TASKING®

EVALUATION GUIDE

VX-TOOLSET

FOR TriCore™

The latest embedded software development toolset for the TriCore™ and AURIX™ microcontroller family offers substantial performance gains over previous generations of TriCore™ development tools. Based on sophisticated Viper compiler technology, this suite of tools has been released as the **TASKING® VX-toolset for TriCore™**. The VX-toolset takes a major step forward by offering unparalleled code optimization performance, advanced multi-core support, an integrated debugger, and integration into the popular Eclipse™ platform as an Integrated Development Environment (IDE).

Eclipse™ integrates the TriCore™ compiler, assembler and linker seamlessly into a single IDE and provides functionality to set up the developer's application and configuration for target debugging. The debugger is integrated into Eclipse™ through a plug-in and offers the user a complete suite of development tools available within a state-of-the-art and industry-standard IDE. This toolset also offers several third-party plugin modules for extended functionality and performance.

The **TASKING VX-toolset** is available in several bundles - **Standard, Professional, Premium and Enterprise Editions** - allowing you to choose the best fit for your application development activities. Along with the standard Eclipse™, C/C++ compiler toolset and simulator modules, additional functionality includes options such as a C compilers for the HSM, GTM, SCR and PCP cores, on-hardware debugging through an OCDS solution and a USB-to-JTAG wiggler.

The included C compilers for the GTM, HSM, SCR and PCP provide a truly unique and complete offering on the market. These compilers maximize the usability of TriCore™ and AURIX™ processors by dealing with the inherent multi-core development opportunities, as well as the limitations of restricted co-processors like the GTM, SCR and PCP, through generation of efficient and compact code.

The OCDS debugger supports debugging using the JTAG bus on a variety of standard evaluation boards from various manufacturers, as well as on custom boards. In addition, third-party debug support is guaranteed by leading tool vendors, such as iSYSTEM®, Lauterbach® and PLS®.

Keeping track of the pin assignments of the AURIX™ can be a tedious task, which is often performed in MS Excel®. TASKING provides the AURIX™ Pin Mapper, through which the internal controls of the MCU can be assigned to the external pins. The Pin Mapper is part of the **AURIX Configuration Tools (ACT)**, which includes an easy to use configurator for Infineon® iLLD drivers as well as a royalty free RTOS, all provided in the **TASKING VX-toolset** through an exclusive partnership with Infineon®.

ECLIPSE™ IDE

The Integrated Development Environment (IDE) that is built on the Eclipse™ framework provides a seamless workbench for the complete tool chain including the debugger of the VX-toolset. The IDE provides facilities for project configuration and management, C/C++ and assembly code-aware editing, build management, debugging, profiling and more. It provides functionality to help you set up your embedded TriCore™ project and configure your target board settings to debug your project on hardware.

The Eclipse™ editor supports C, C++, assembly language and header files with syntax highlighting, auto completion, context assistance and tool tips. As you would expect from a de facto standard IDE, it provides full support for all relevant source code version control systems. The Eclipse™ environment provides a single platform for many different embedded product toolsets from different vendors. The standardization on an industry-wide IDE significantly reduces your learning curve, removes the barriers of changing development tools for different architectures, increases your productivity, and ultimately reduces the time to market with your end product. The availability of plugin modules to enhance or extend the feature set of the Eclipse™ IDE ensures that you can build the workbench according to your development needs. With the concept of the open Eclipse™ framework, third-party tool vendors can now develop plugins that tightly integrate into various IDEs from different vendors, unlike proprietary IDEs where custom connectivity needs to be created.

The VX-toolset's IDE is based on the Luna release of Eclipse™ and the C/C++ Development Tools (CDT). TASKING has built the integration blocks for the toolset and extensions to Eclipse™ to make the whole environment a coherent workbench.

C COMPILER

Based upon the latest compiler technologies, all VX-toolset C compilers are reliable, compliant, competitive, complete, compatible and easy to use to generate the most optimal code possible. The TASKING VX-compilers are tested for ISO C'99 and ISO C++ conformity against authoritative validation suites, such as Perennial® and Plum Hall®. In addition, the optimization techniques of the compilers are tested with large real-world applications (for example, powertrain and body control sources), as well as industry benchmark standards such as Nullstone and EEMBC.

Fast and Compact

TASKING understands that you expect your TriCore™ compiler to produce the most optimal code possible with hassles. With its Viper compiler technology, the TASKING VX-toolset for TriCore™ generates code with the smallest footprint and fastest execution possible. Depending on the specific requirements of your TriCore™ application, optimizations can be further tweaked for smaller code sizes or higher execution speeds.

Compiler optimizations include:

- Partial Redundancy Elimination (PRE) detects and eliminates repeating (sub-) expressions.
- Various Loop and Jump optimizations speed up execution and reduce code size.
- Control-low and code-reduction optimizations remove dead code and perform transformations to minimize jumps.
- Function inlining replaces calls to small functions with inlined copies of the function code.
- Peephole optimizations replace instruction sequences with equivalent but faster and/or shorter sequences, or remove obsolete instructions.
- Inter-procedural register allocation
- Application wide code compaction (or reverse inlining)

Code Profiling

In addition to the profiling features built into the debugger, the compiler is equipped with a profiler that uses code instrumentation. Code profiling can be used to determine which pieces of your code execute slower than expected and which functions contribute to the overall execution time of a program. A profile can also tell you which functions are called more or less often than expected. The advantage of this code profiling option in the compiler is that it can give a complete call graph of the application annotated with the time spent in each function and basic block.

Syntax and Semantic Checks

The compiler offers a vast array of syntax and semantic checks that warn about potential undesirable effects or bugs in your program. Early fixing of source code problems when reported by the compiler generally only takes minutes compared to hours, or days, when the problem is discovered at run time.

Examples of compile-time checks includes:

- Validating ***printf*** and ***scanf*** format strings against the type of the actual arguments
- Using uninitialized memory locations
- Detecting unused variables

- Value tracking, which is used to detect errors such as:
  - Array subscript out of bounds
  - Division by zero
  - Constant conditions

Run-time Error Checking

The TASKING VX-toolset's run-time error checking capabilities in the compiler offer a wealth of checks that reveal run-time errors when they first occur. The kind of errors found by run-time error checking are typically hard to find since they manifest themselves through secondary effects or, in the worst case, will not manifest at all prior to your product being shipped. By identifying the source line where the error first occurs, the run-time error checking facilities reduce the time spent in the debugger and increase the quality of your software. You can specify whether the application will terminate or continue when an error is detected.

These optional checks are implemented by generating additional code and/or enabling additional code in the standard C library. Run-time error checking has a nominal effect on code size and execution speed and can be enabled on a module-by-module basis, making it practical for use in debugging large applications.

The following types of checks are provided:

- Bounds checking verify all pointer operations to detect buffer overflows and other illegal operations including:
  - Accessing uninitialized or null pointers
  - Accessing objects outside their declared bounds
  - Illegal pointer arithmetic
- Malloc / free checks uncover dynamic memory allocation errors including:
  - Buffer overflow
  - Write to freed memory
  - Multiple calls to free
  - Passing an invalid pointer to free
- Report an unhandled case value in a switch without a default part.
- Stack overflow detects when the stack grows beyond its allocated size.
- Divide by zero issues a message when a division by zero is attempted.

CPU Functional Problem Support

Semiconductor vendors regularly publish microcontroller errata sheets reporting deviations from the electrical and timing specifications. As an integral part of best practice architecture support, the TASKING VX-toolset for TriCore™ provides bypasses and checks for identified silicon defects.

CPU functional problem support is provided throughout the complete toolset, including:

- C-compiler bypasses adapt code generation in order to avoid the identified erratic instruction sequences.
- Assembler checks warn the assembly programmer for suspicious or erroneous instruction sequences.
- Protected C-library sets are built with bypasses for all identified CPU functional problems.

If reliability of your embedded application is essential, be sure to put support for CPU functional problems on your list of compiler selection criteria. Through its close partnership with semiconductor vendors, TASKING offers the most comprehensive support for this with its TASKING® compilers.

COMPILER CHECKLIST		
Components tightly integrated in a single IDE	**TASKING**	**COMPETITION**
TriCore™ compiler	✓	✓
GTM compiler	✓	✗
HSM compiler	✓	✗
SCR compiler	✓	✗
Eclipse™ integrated debugger	✓	✗
AURIX™ Pin Mapper	✓	✗
Infineon® iLLD configurator	✓	✗

STATIC CODE ANALYSIS

Static code analysis is a method used to verify all possible paths within a software program without actually executing the program. A static code analysis tool can efficiently locate defects including out of bound array access, memory allocation errors, arithmetic over and underflows, and inconsistent code fragments that go unnoticed during dynamic tests or peer reviews. Static code analysis can be applied early in the software development process, and can be applied on incomplete and incorrect code bases and when no test-cases need to be developed.

TASKING has integrated static code analysis functionality for **CERT C** and **MISRA C** in its C compilers, with the advantage that such an analyzer is aware about specific embedded software issues such as: the existence of special function registers, the use of in-line assembly language, C-language extensions such as pointer and memory space qualifiers to address multiple address spaces, and DSP specific data types such as circular buffers, and fixed point data types.

Integrated Safety Checker

The latest release of the VX-toolset for TriCore™ has been enhanced with a **Code Safety Checker** functionality, which helps the developer to prevent interference between software elements with incompatible **ASIL** levels in an application. Unlike traditional static analysis tools that check for dangerous code constructs against the specification, the **Code Safety Checker** operates closely to the hardware while still able to check high-level safety requirements.This makes it an ideal companion, bridging the gap between **ISO 26262** requirements and traditional software tests. The **Code Safety Checker** allows you to assign linker sections to safety classes you define, even for existing object files where the source code isn't available. This information is added to the ELF file as debug information, so the application won't be modified. Access between different safety classes and violations will be shown immediately.

CERT C

The **CERT C/C++** secure coding standard is defined by the **Computer Emergency Readiness Team (CERT)**, founded by the US government. TASKING is one of the first vendors to provide a **CERT C** coding guidelines analyzer built into a C compiler for embedded software development.

MISRA C

MISRA C is driven by the **Motor Industry Software Reliability Association** and guides programmers in writing more robust C-code by defining selectable C-usage restriction rules. Through a system of strict error checking, the use of error-prone C-constructs can be prevented. The TASKING C compiler offers the industry's first support for **MISRA-C:1998**, **MISRA-C:2004** and the latest **MISRA-C:2012** guidelines.

C COMPILER FOR GTM

Next to TASKING's standalone C compiler for the **Generic Timer Module (GTM)**, the TASKING TriCore™ toolset includes a fully integrated C compiler that supports the third generation **GTM** core from Bosch®. Programming a complex core like the **GTM** in C makes you more productive both in the development phase as well as the maintenance stage. The first **GTM** core generation present on the AURIX™ variants is fully supported through the included **MCS/GTM** assembler.

C COMPILER FOR HSM

For programming the HW Security Module of the AURIX™ microcontrollers, the TASKING VX-toolset includes a fully integrated and dedicated C compiler. This compiler is based on the standard C compiler for the Cortex-M series from TASKING, and it can easily be accessed from the TriCore™ toolset's IDE.

C COMPILER FOR SCR

The AURIX TC3xx Family as well as select AURIX™ derivatives, such as the TC26x series, do have a dedicated 8-bit Standby Controller (SCR) on board. This controller is based on Infineon® XC800 microcontroller and TASKING has developed a brand-new and highly optimized C compiler to program this core. This exclusive compiler based on VX technology from TASKING, generates more efficient code than traditional XC800 compilers and is therefore the ultimate programming tool to deal with the limited resources of the Standby Controller. It is fully integrated in the VX-toolset for TriCore™ and does not require a third-party compiler solution.

TARGET ARCHITECTURE SUPPORT

The TASKING VX-toolset supports all TriCore™ derivatives. From within the Eclipse IDE you can easily select the TriCore™ device of your choice for your project:

- Devices based on AURIX™: TC21x, TC22x, TC23x, TC23x_ ADAS, TC26x, TC27x, TC29x
- AURIX TC3xx Family
- TriCore™ devices: TC1130, TC1166, TC1167, TC1184, TC1197, TC1337, TC1367, TC1387, TC1736, TC1746 [TC1782bd], TC1724, TC1728, TC1748 [TC1798bd], TC1762, TC1764, TC1766, TC1767, TC1768 [TC1387bd], TC1782, TC1791, TC1792, TC1793, TC1796, TC1798

TASKING's active relationship with Infineon® Technologies enables us to support new derivatives already in the toolset prior to their availability in volume.

C COMPILER FOR PCP

TASKING is offering a unique C compiler for TriCore™ **Peripheral Control Processor (PCP)**. Despite the limited functionality and restricted instruction set of the **PCP**, we have been able to develop a fully functional C compiler. The C compiler delivers code at an unexpectedly high performance level and provides several special extensions for **PCP** programming.

MULTI-CORE SUPPORT

The VX-toolset provides two advanced methods for multi-core program development for the AURIX™ microcontroller, which includes:

- **Compile time core association:** Crosscore access is detected by the compiler in an early stage. This allows for the best control of code and data destination on the various cores. Stricter coding discipline is asked from the developer, resulting in better quality control while delivering optimal program execution.
- **Link time core association:** The destination core is selected in the final step, delivering ultimate programming flexibility and allowing for straightforward reusability of existing code.

For ultimate flexibility and program execution, both methods can be used together.

INTEGRATED DEBUGGER

The VX-toolset's debugger is based on the latest debugger technologies from TASKING. This debugger has been redesigned from the ground up and is ready for market trends like integrated kernel-awareness and multi-core debugging. Utilizing the Eclipse™ IDE workbench, it comes as a plug-in with a seamless integration to the editing environment. With the VX-toolset for TriCore™ it provides two execution environments serving various debugging needs.

OCDS Debugging Through Infineon® DAS Support

Making the most of the On-Chip-Debug-Support (OCDS) facilities built into the Infineon® TriCore™ microcontrollers, our debugger offers an accessible, high quality in-circuit-emulation functionality. The VX-toolset has been tested and qualified with **Infineon® Debug Access Server (DAS)** solution. The **DAS** environment is the universal emulation access software architecture for all Infineon® microcontroller families. Extensive support for **DAS** is guaranteed by Infineon® and, as a result, TASKING has adopted this debug standard.

Through **DAS**, the TASKING TriCore™ debugger is compatible with Infineon® starter kits with an on-board wiggler through a parallel or USB cable. This debugger is also compatible with **Infineon® DAS miniWiggler**, enabling a very cost-effective debug solution for on-hardware testing for custom hardware or other evaluation boards.

TriCore™ Instruction Set Simulator Debugging

The TriCore™ simulator debugger features instruction set simulation, allowing you to extensively debug your application on the host platform, even before your target hardware is available. A plug-in for instruction set simulation of the **PCP** is included.

AURIX™ CONFIGURATION TOOLS

Advanced microcontrollers are equipped with a large number of on-chip peripheral modules, but the limited number of pins on the chip usually does not allow all modules to be used simultaneously. TASKING Pin Mapper functionality removes the developer's complex challenge of configuring chip hardware registers that are used for assigning the peripheral module signals to the physical pins. The Pin Mapper provides an interactive visual representation of the pin layout within the toolset IDE, through which the developer can configure and review properties of the pins. The Pin Mapper reports errors or warnings for possible connection conflicts, saving the developer from the tedious task of maintaining an overview of the pin assignments in spreadsheets.

Through the graphical editor and code generator from TASKING's award winning **Software Platform**, you can quickly configure **(Infineon® iLLD)** low-level drivers, various C files and header files, as well as the RTOS for use in final product applications. This significantly simplifies the required steps to program an advanced microcontroller.

WHICH BUNDLE IS THE BEST FOR YOU?

The TASKING VX-toolset is available in targeted bundles - **Standard, Professional, Premium and Enterprise Editions**, allowing you to choose the best fit for your application development activities.

The **Enterprise Edition** is the best choice if you want to have all essential development tools around the C/C++ compiler integrated into one environment. In addition to a software simulator, it offers an on-hardware debugging solution – using an On-Chip Debug System (OCDS) solution with a USB-to-JTAG wiggler. The OCDS debugger is the most cost-effective on-hardware debug solution you can get. The truly unique parts of the Enterprise Edition are the C compilers for the **HW Security Module (HSM)**, the **Generic Timer Module (GTM)**, the **Standby Controller (SCR)** and the **Peripheral Control Processor (PCP)**, as well as the **Code Safety Checker** for **ASIL**. If you plan to write your code for the additional cores in C language, this **Enterprise Edition** is the best and only option on the market.

The Premium Edition is nearly as comprehensive, with the exception of the **GTM** compiler and the integrated **Code Safety Checker**.

The **Professional Edition** provides many features of the **Premium Edition,** including the OCDS debugger and **HSM** C compiler, but without the **SCR** and **PCP** C compilers. If you develop your application based on an AURIX™ with **HSM** co-processor, or a TriCore™ derivative without the **PCP**, this **Professional Edition** is an attractive solution to consider. Also, if you are fine with programming your TriCore™ with **PCP** in assembly code instead of C, this package is a good choice. If you purchase this bundle, be sure to add the optional USB-to-JTAG miniWiggler in case your hardware board comes without an on-board wiggler.

The **Standard Edition** is the perfectly bundle for C/C++ programming and debugging with a simulator. The **MCS/GTM** unit of the AURIX™ can be programmed through the included assembler, whereas the linker is fully prepared for multi-core TriCore™ development. You can upgrade to the one of the other Editions at a later time, offering you all included functionality under a unified interface.